OpenGazeLab: An Interactive Toolkit for Gaze Analysis and Event Detection

Khue Minh Pham*
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
Saarland Informatics Campus
Saarland University
Saarbrücken, Germany
minh_khue.pham@dfki.de

Abdulrahman Mohamed Selim*
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
abdulrahman.mohamed@dfki.de

Omair Shahzad Bhatti
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
omair_shahzad.bhatti@dfki.de

László Kopácsi
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
laszlo.kopacsi@dfki.de

Michael Barz
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
michael.barz@dfki.de

Daniel Sonntag
Interactive Machine Learning
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
Applied Artificial Intelligence
University of Oldenburg
Oldenburg, Germany
daniel.sonntag@dfki.de

Abstract

Event detection (e.g., fixations and saccades) is a prerequisite for many eye-tracking analyses. However, existing solutions often require coding experience, rely on closed-source vendor tools, or focus on narrow paradigms (e.g., reading). Additionally, most are designed for stationary eye tracking, whereas head-mounted recordings are affected by head/scene motion, making event detection more difficult. Therefore, we present OpenGazeLab, a publicly available, browser-based toolkit that unifies event extraction, parameter configuration, and data inspection for both stationary and head-mounted eye tracking. OpenGazeLab implements I-DT and I-VT, and extends them for head-mounted data using scene-motion compensation and adaptive thresholds. The toolkit also provides a timeline-based visualisation that overlays gaze and detected events on the stimulus image or scene video, enabling quick visual verification. OpenGazeLab is implemented using widely used, well-maintained Python frameworks to support reproducible, easy-to-adopt workflows, and we plan to extend it with additional event classes (e.g., smooth pursuit) and alternative detectors.

CCS Concepts

• **Human-centered computing → Human computer interaction (HCI)**; *Web-based interaction*; *Visualization toolkits.*

*These authors contributed equally to this research.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ETRA '26, Marrakesh, Morocco
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2519-7/26/06
https://doi.org/10.1145/3797246.3805855

Keywords

Eye Movement Event Detection, Interactive User Interface, Data Visualisation

ACM Reference Format:
Khue Minh Pham, Abdulrahman Mohamed Selim, Omair Shahzad Bhatti, László Kopácsi, Michael Barz, and Daniel Sonntag. 2026. OpenGazeLab: An Interactive Toolkit for Gaze Analysis and Event Detection. In *2026 Symposium on Eye Tracking Research and Applications (ETRA '26), June 01–04, 2026, Marrakesh, Morocco.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3797246.3805855

1 Introduction

Eye-tracking is widely used across multiple domains, e.g., psychology, human-computer interaction, and education, in order to get insights into human cognition and visual attention [Bhatti et al. 2024; Fu et al. 2024; Holmqvist et al. 2011; Mohamed Selim et al. 2024a,b]. In practice, many analyses depend on transforming raw gaze samples into oculomotor events, such as fixations and saccades, often using methods like Dispersion-Threshold Identification (I-DT) and Velocity-Threshold Identification (I-VT) [Salvucci and Goldberg 2000]. However, implementing these methods can be challenging for users with limited programming experience, as it typically involves writing custom code, handling noisy data, and tuning parameters to reliably detect events. These challenges are amplified in head-mounted settings, where head motion and dynamic scenes can reduce gaze stability relative to the environment, making event detection more difficult. In addition, manufacturer-provided detection algorithms are often device-specific and limit the ability to inspect, interpret, and fine-tune the underlying methods [Lappi 2015].

Although several open-source tools support gaze processing, they involve different trade-offs. For example, *CatEyes* [Gütlin 2021]

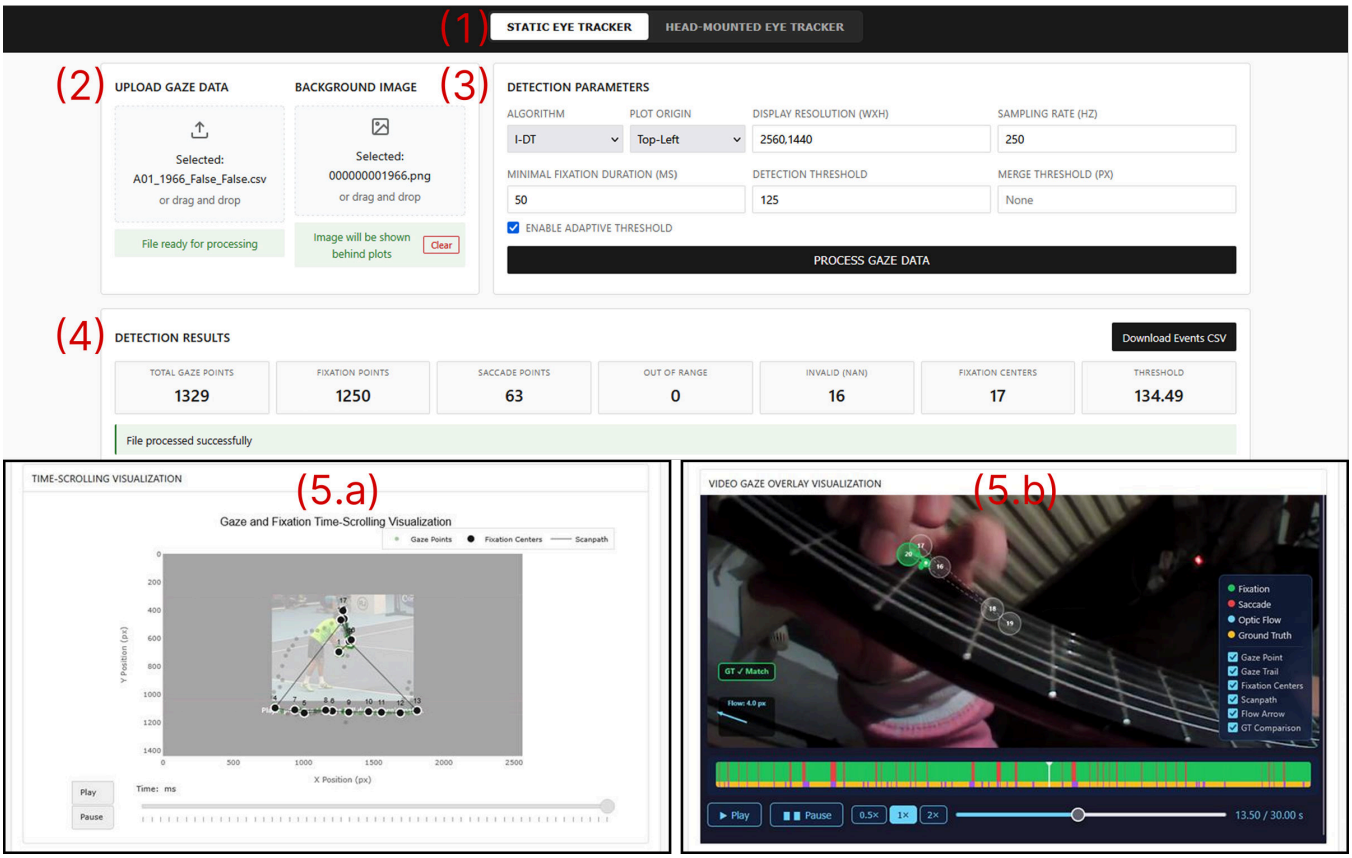

Figure 1: A screenshot of the user interface from OpenGazeLab. The gaze data visualisation in (5.a) is on a stimulus from the FOIL-COCO dataset [Shekhar et al. 2017], and in (5.b) is from the dataset of Drews and Dierkes [2024].

provides a general-purpose processing toolbox, but is not actively maintained; *pymovements* [Krakowczyk et al. 2023] offers various processing methods, but requires coding experience and lacks a user interface (UI); and *Eyekit* [Carr 2025] provides strong support for reading analysis, but is primarily designed around text-based paradigms. These tools are valuable, but there is still a need for an easy-to-use toolkit that supports event detection and interactive, in-context verification for both stationary and head-mounted eye tracking, and integrates with standard Python frameworks.

To address this gap, we present OpenGazeLab, a browser-based toolkit for eye-tracking event detection and visual verification[1]. Our contributions are:

(1) Interactive UI for event detection, enabling users to load gaze data, configure event-detection parameters, and export detected events.
(2) Implementing standard I-DT and I-VT and adapting them for head-mounted settings based on the framework of Drews and Dierkes [2024].
(3) In-context visual verification using a timeline slider and synchronised video playback that overlays gaze and detected events on stimulus images or scene videos for visual inspection.

2 System Overview

The OpenGazeLab UI (Figure 1) provides a simple workflow for processing gaze recordings and inspecting the results in context.

2.1 Event Detection Algorithms

OpenGazeLab currently supports two widely used event-detection algorithms, i.e., I-DT and I-VT [Salvucci and Goldberg 2000], which are commonly applied to remote, stationary eye trackers. However, applying threshold-based detectors in mobile settings is more challenging, because stimulus dynamics and head/scene motion can introduce additional artefacts and increase sensitivity to parameter choices (e.g. threshold value) [Andersson et al. 2017; Startsev and Zemblys 2023]. Therefore, for head-mounted eye tracking, OpenGazeLab follows the three-step framework proposed by Drews and Dierkes [2024]: (1) it compensates for head motion by estimating global scene motion via sparse Lucas-Kanade optical flow [Lucas and Kanade 1981] and subtracting it to approximate relative gaze velocity; (2) it applies adaptive thresholds scaled by

[1]Our codebase is publicly available on GitHub.

the Root Mean Square (RMS) amplitude of the optical flow to account for real-time head movement intensity; and (3) it applies post-processing optimisation (e.g., minimum fixation duration and microsaccade-based merging) to reduce event fragmentation.

2.2 Preprocessing and Parameter Tuning

OpenGazeLab currently includes optional methods for preprocessing and automatic parameter suggestions to reduce manual trial-and-error. When processing is triggered, the toolkit can apply a *Savitzky-Golay filter* [Savitzky 1989; Savitzky and Golay 1964] to smooth raw coordinates while preserving fast saccade transitions. It also performs *temporal segmentation* using a dynamic gap threshold to isolate data gaps (e.g., blinks or tracking loss) and avoid artificial interpolation. In addition, OpenGazeLab can suggest detection thresholds in two stages: first, it scales baseline thresholds using the *Median Absolute Deviation (MAD)* of gaze velocity to account for participant- and recording-specific noise; second, it searches a range of thresholds and selects values that maximise the *Calinski-Harabasz (CH) score* [Caliński and Harabasz 1974] under this objective. These steps are optional and can be toggled on and off depending on the dataset and analysis goals.

2.3 Implementation

To support cross-platform use, OpenGazeLab uses a lightweight client-server architecture. The frontend is built with React, while the backend is implemented in Python with FastAPI. For gaze processing, the backend relies on standard, well-maintained packages such as pandas, NumPy, scikit-learn, and SciPy. In addition, for head-mounted setups, OpenCV is used to extract scene-video metadata (e.g., frame rate and resolution) from uploaded videos. Finally, interactive visualisations are rendered in the browsers using Plotly.

3 Interface and Workflow

As previously mentioned, the UI, as shown in Figure 1, follows a simple workflow. After selecting the eye-tracking setup (Step 1 in Figure 1), the user proceeds through four steps[2].

3.1 Upload and Initialisation

OpenGazeLab supports raw gaze CSV formats (Step 2 in Figure 1). After a drag-and-drop upload, the backend parses the file by detecting the delimiter, mapping the spatial and temporal columns, and checking whether the coordinates are normalised or pixel-based. Users can optionally upload a stimulus image or scene video, which is used as a background for in-context visual inspection.

3.2 Detection Parameters

Users configure event detection through a compact control panel (Step 3 in Figure 1). They select the detection method (I-DT or I-VT) and set core parameters, including sampling rate, display resolution, minimum fixation duration, and the coordinate origin. Optional checkboxes enable or disable adaptive thresholds and automatic parameter suggestions before running the pipeline.

[2]A demo video is available online: YouTube Video

3.3 Detection Results

After processing, the interface shows a concise summary of the output (Step 4 in Figure 1). This includes the number of detected events (e.g., fixations and saccades) and basic data-quality indicators (e.g., out-of-range samples and missing values). The interface also reports the selected threshold values (if automatic suggestions are enabled), and users can export the event list as a CSV file for downstream analysis.

3.4 Timeline-based Visualisation

Finally, OpenGazeLab supports interactive inspection using timeline-based visualisation. **For stationary/remote eye trackers**, users can scroll through the timeline using a slider (and optional playback controls) while viewing gaze and detected events overlaid on a stimulus image or on a standard plot when no stimulus is provided (Step 5.a in Figure 1). **For head-mounted eye trackers**, the toolkit provides synchronised scene-video playback with overlays of gaze and detected events (Step 5.b in Figure 1). Users can toggle multiple visual layers, including gaze points and trails, duration-scaled fixation centres, connected scanpaths, optical-flow motion cues, and optional ground-truth comparisons (when available). A clickable event timeline and playback controls support navigation through the recording. Together, these views support rapid visual checking of event labels and help users iterate on parameters when needed.

4 Conclusion, Limitations, and Future Work

In this paper, we presented OpenGazeLab, a browser-based toolkit that supports event detection and in-context inspection for both stationary and head-mounted eye tracking. The toolkit combines configurable threshold-based detectors with interactive visual overlays to help researchers process data and quickly inspect detected events. The codebase is publicly available on GitHub to support reproducible and extensible workflows.

OpenGazeLab also has certain limitations. Our evaluation so far relies mainly on publicly available, but internally collected, stationary datasets [Barz et al. 2022; Bhatti et al. 2024; Mohamed Selim et al. 2026] using Tobii eye trackers, and a small number of external head-mounted datasets [Drews and Dierkes 2024; Kothari et al. 2020]. In addition, the current event detection is limited to threshold-based methods and does not yet include additional event classes (e.g., smooth pursuit) or machine-learning-based detectors [Hoppe and Bulling 2016; Nejad et al. 2024; Startsev et al. 2019]. Finally, the preprocessing and filtering options are currently limited.

We plan to address these limitations in future work by validating the toolkit across a broader range of datasets and recording conditions, extending it beyond threshold-based event detection with additional event classes and alternative detectors, and expanding the available preprocessing options. We also aim to add event-based feature extraction to better support downstream analyses. Finally, inspired by *Eyekit* [Carr 2025], we aim to explore automated Area of Interest (AOI) detection and mapping for mixed-media stimuli in a way that remains general (i.e., not limited to text-only paradigms) and easy to inspect and refine.

Acknowledgments

This work was funded by the European Union under grant number 101093079 (MASTER), the Federal Ministry of Research, Technology and Space (BMFTR) under grant number 16IW23002 (No-IDLE) and grant number 16IW24006 (NoIDLEChatGPT), the Lower Saxony Ministry of Science and Culture (MWK) in the zukunft.niedersachsen program, and the Endowed Chair of AAI at the University of Oldenburg.

References

Richard Andersson, Linnea Larsson, Kenneth Holmqvist, Martin Stridh, and Marcus Nyström. 2017. One Algorithm to Rule Them All? An Evaluation and Discussion of Ten Eye Movement Event-Detection Algorithms. *Behavior Research Methods* 49, 2 (April 2017), 616–637. https://doi.org/10.3758/s13428-016-0738-9

Michael Barz, Omair Shahzad Bhatti, and Daniel Sonntag. 2022. Implicit Estimation of Paragraph Relevance From Eye Movements. *Frontiers in Computer Science* 3 (Jan. 2022). https://doi.org/10.3389/fcomp.2021.808507 Publisher: Frontiers.

Omair Shahzad Bhatti, Harshinee Sriram, Abdulrahman Mohamed Selim, Cristina Conati, Michael Barz, and Daniel Sonntag. 2024. Detecting when Users Disagree with Generated Captions. In *Companion Proceedings of the 26th International Conference on Multimodal Interaction*. ACM, San Jose Costa Rica, 195–203. https://doi.org/10.1145/3686215.3688382

T. Caliński and J Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics* 3, 1 (Jan. 1974), 1–27. https://doi.org/10.1080/03610927408827101 Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/03610927408827101.

Jon Carr. 2025. jwcarr/eyekit. https://github.com/jwcarr/eyekit original-date: 2019-12-13T16:07:15Z.

Michael Drews and Kai Dierkes. 2024. Strategies for enhancing automatic fixation detection in head-mounted eye tracking. *Behavior Research Methods* (April 2024). https://doi.org/10.3758/s13428-024-02360-0

Xiaoxue Fu, John M. Franchak, Leigha A. MacNeill, Kelley E. Gunther, Jeremy I. Borjon, Julia Yurkovic-Harding, Samuel Harding, Jessica Bradshaw, and Koraly E. Pérez-Edgar. 2024. Implementing mobile eye tracking in psychological research: A practical guide. *Behavior Research Methods* 56, 8 (Dec. 2024), 8269–8288. https://doi.org/10.3758/s13428-024-02473-6

Dirk Gütlin. 2021. CatEyes. https://github.com/DiGyt/cateyes original-date: 2021-08-28.

Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, Joost van de Weijer, Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost van de Weijer. 2011. *Eye Tracking: A comprehensive guide to methods and measures*. Oxford University Press, Oxford, New York.

Sabrina Hoppe and Andreas Bulling. 2016. End-to-End Eye Movement Detection Using Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1609.02452 arXiv:1609.02452 [cs].

Rakshit Kothari, Zhizhuo Yang, Christopher Kanan, Reynold Bailey, Jeff B. Pelz, and Gabriel J. Diaz. 2020. Gaze-in-Wild: A Dataset for Studying Eye and Head Coordination in Everyday Activities. *Scientific Reports* 10, 1 (Feb. 2020), 2539. https://doi.org/10.1038/s41598-020-59251-5

Daniel G. Krakowczyk, David R. Reich, Jakob Chwastek, Deborah N. Jakobi, Paul Prasse, Assunta Süss, Oleksii Turuta, Paweł Kasprowski, and Lena A. Jäger. 2023. pymovements: A Python Package for Eye Movement Data Processing. In *Proceedings of the 2023 Symposium on Eye Tracking Research and Applications (ETRA '23)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/3588015.3590134

Otto Lappi. 2015. Eye Tracking in the Wild: The Good, the Bad and the Ugly. *Journal of Eye Movement Research* 8, 5 (Oct. 2015), 1–21. https://doi.org/10.16910/jemr.8.5.1 Company: Bern Open Publishing Distributor: Bern Open Publishing Institution: Bern Open Publishing Label: Bern Open Publishing Publisher: publisher.

Bruce D. Lucas and Takeo Kanade. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2 (IJCAI'81)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 674–679. https://www.ijcai.org/Proceedings/81-2/Papers/017.pdf

Abdulrahman Mohamed Selim, Michael Barz, Omair Shahzad Bhatti, Hasan Md Tusfiqur Alam, and Daniel Sonntag. 2024a. A review of machine learning in scanpath analysis for passive gaze-based interaction. *Frontiers in Artificial Intelligence* 7 (June 2024). https://doi.org/10.3389/frai.2024.1391745 Publisher: Frontiers.

Abdulrahman Mohamed Selim, Omair Shahzad Bhatti, Michael Barz, and Daniel Sonntag. 2024b. Perceived Text Relevance Estimation Using Scanpaths and GNNs. In *International Conference on Multimodel Interaction*. ACM, San Jose Costa Rica, 418–427. https://doi.org/10.1145/3678957.3685736

Abdulrahman Mohamed Selim, Omair Shahzad Bhatti, Amr Gomaa, Michael Barz, and Daniel Sonntag. 2026. Do You (Dis)agree With Me? Modelling Implicit User Disagreement in Human–AI Interaction Using Gaze Data. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery (ACM), Barcelona, Spain. https://doi.org/10.1145/3772318.3790594

Ashkan Nejad, Gera A. De Haan, Joost Heutink, and Frans W. Cornelissen. 2024. ACE-DNV: Automatic Classification of Gaze Events in Dynamic Natural Viewing. *Behavior Research Methods* 56, 4 (March 2024), 3300–3314. https://doi.org/10.3758/s13428-024-02358-8

Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on Eye tracking research & applications (ETRA '00)*. Association for Computing Machinery, New York, NY, USA, 71–78. https://doi.org/10.1145/355017.355028

Abraham Savitzky. 1989. A Historic Collaboration. *Analytical Chemistry* 61, 15 (Aug. 1989), 921A–923A. https://doi.org/10.1021/ac00190a744 Publisher: American Chemical Society.

Abraham. Savitzky and M. J. E. Golay. 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry* 36, 8 (July 1964), 1627–1639. https://doi.org/10.1021/ac60214a047 Publisher: American Chemical Society.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. FOIL it! Find One mismatch between Image and Language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 255–265. https://doi.org/10.18653/v1/P17-1024

Mikhail Startsev, Ioannis Agtzidis, and Michael Dorr. 2019. 1D CNN with BLSTM for Automated Classification of Fixations, Saccades, and Smooth Pursuits. *Behavior Research Methods* 51, 2 (April 2019), 556–572. https://doi.org/10.3758/s13428-018-1144-2

Mikhail Startsev and Raimondas Zemblys. 2023. Evaluating Eye Movement Event Detection: A Review of the State of the Art. *Behavior Research Methods* 55, 4 (June 2023), 1653–1714. https://doi.org/10.3758/s13428-021-01763-7